DST 2 IPT

Exercice 1 - Quelques fonctions en python.

1. Vous avez oublié quelle commande python renvoie la valeur absolue. Proposer une fonction *valeur_ abs(x)* qui renvoie la valeur absolue de x.
2. Etant donnée une liste L de nombres réels, proposer une fonction *mon_ min(L)* qui renvoie son minimum.
3. Proposer une fonction *change_ point(chaine)* qui prend en argument une chaîne, et qui remplace toutes les virgules par des points.
   On rappelle qu'une chaîne n'est pas dynamique, c'est-à-dire ne peut pas être modifiée en place : il faudra créer une nouvelle chaîne.
4. Soit la suite définie par $u_{n+1} = 2u_n(1-u_n)$ et $u_0 = \frac{1}{4}$.
   Ecrire une fonction *calcul(N)* qui calcule les termes de la suite jusqu'à u_N, et renvoie la liste des valeurs de la suite, de u_0 jusqu'à u_N.
5. Pour la même suite, écrire une deuxième fonction *calcul_ 2(r)*, qui dépend d'un réel $r > 0$, et qui calcule les termes de la suite jusqu'à ce que $|u_{n+1} - u_n| < r$, ou bien qui s'arrête à la centième itération si le critère d'arrêt $|u_{n+1} - u_n| < r$ n'a jamais été vérifié. La fonction renverra le dernier terme calculé.

Exercice 2 - Représentation des nombres.

1. Donner (en détaillant) la représentation en base 10 des nombres suivants :
   - **a.** Le nombre $(10100101)_2$.
   - **b.** Le nombre $(1A3)_{16}$.
2. Donner (en détaillant) la représentation en base 2 des nombres suivants :
   - **a.** Le nombre $(14)_{10}$.
   - **b.** Le nombre $(315)_{10}$.
   - **c.** Le nombre $(28)_{16}$.
3. On rappelle qu'en représentation en complément à 2 sur n bits (complément à 2^n pour être exact !), on peut représenter tous les nombres entiers de l'intervalle $[-2^{n-1}, 2^{n-1}-1]$: si un nombre x vérifie $0 \leqslant x \leqslant 2^{n-1}-1$, on le code normalement en binaire, sinon on le code par la représentation binaire de $x + 2^n$.
   - **a.** Coder sur 8 bits, en complément à deux, les nombre $x = 73$ et $y = -89$
   - **b.** Donner la valeur des entiers relatifs suivants, codés sur 6 bits en complément à deux : $M = (101010)_2$ et $N = (010101)_2$.
4. Dans cet exercice, on considérera des nombres flottants, donnés par leur représentation par signe, exposant et mantisse. On utilise ici une représentation de flottants sur 9 bits « maison » (ça fait moins de chiffres !)
   - Le premier bit donne le signe,
   - Les 4 suivants donnent l'exposant e, et l'exposant final est donné par $E = e - E_{max}$ où $E_{max} = 2^3 - 1$.
   - Les 4 suivants donnent la mantisse, qui se calcule en puissances négatives de 2 de manière standard. Elle est comprise entre 1 et 2 grâce à son « bit caché ».
     La formule finale est celle habituelle :
     $$x = (-1)^s \times m \times 2^E.$$
   - **a.** Quel est le plus grand réel que l'on peut coder avec cette représentation ?
   - **b.** Que vaut le nombre $A = 101011000$ dans cette représentation ?

Correction :

1. On renvoie au cours pour les métodes détaillées, ces questions sont toutes élémentaires

a. Le nombre comporte 8 bits, on a $(10100101)_2 = 1 + 4 + 32 + 128 = 165$.

b. Rappelons que le A représente le chiffre 10 en base 16 : $(1A3)_{16} = 3+10\times 16+1\times 16^2 = 3+160+256 = 419$.

2. **a.** On a : $(14)_{10} = 8 + 4 + 2 = (1110)_2$.

b. Déjà, $256 = 2^8 < 315 < 2^9 = 512$, donc on code le nombre sur 9 bits. On a

$$315 = 256 + 59 = 256 + 32 + 27 = 256 + 32 + 16 + 11 = 256 + 32 + 16 + 8 + 2 + 1.$$

Ainsi :

$$(315)_{10} = (100111011)_2.$$

c. On convertit 2 et 8 en base 2, sur 4 bits : $2 = (0010)_2$ et $8 = (1000)_2$, et on concatène les deux nombres : $(28)_{16} = (00101000)_2$.

On pouvait bien sûr passer par la base 10, mais c'est beaucoup plus long.

3. On rappelle qu'en représentation en complément à 2 sur n bits (complément à 2^n pour être exact !), on peut représenter tous les nombres entiers de l'intervalle $[-2^{n-1}, 2^{n-1} - 1]$: si un nombre x vérifie $0 \leqslant x \leqslant 2^{n-1} - 1$, on le code normalement en binaire, sinon on le code par la représentation binaire de $x + 2^n$.

a. Comme $0 \leqslant 73 \leqslant 2^7 = 128$, on le code normalement en base 2 sur 8 bits : $x = (01001001)$. Le nombre y étant négatif, on le décale en calculant $y + 2^8 = -89 + 256 = 167$, et on le code sur 8 bits : (10100111). On pouvait aussi utiliser la méthode d'inversion.

b. Puisque le nombre M commence par 1, il est négatif, et on le retrouve en convertissant, puis en décalant. En base 2 standard, on a $(101010)_2 = 42$, puis $M = 42 - 2^6 = 42 - 64 = -22$.

Pas de difficulté pour N car il commence par 0, et donc $N = (010101)_2 = 21$.

4. Dans cet exercice, on considérera des nombres flottants, donnés par leur représentation par signe, exposant et mantisse. On utilise ici une représentation de flottants sur 9 bits « maison » (ça fait moins de chiffres !)

- Le premier bit donne le signe,
- Les 4 suivants donnent l'exposant e, et l'exposant final est donné par $E = e - E_{max}$ où $E_{max} = 2^3 - 1$.
- Les 4 suivants donnent la mantisse, qui se calcule en puissances négatives de 2 de manière standard. Elle est comprise entre 1 et 2 grâce à son « bit caché ».

  La formule finale est celle habituelle :

$$x = (-1)^s \times m \times 2^E.$$

a. La valeur maximale de e est $e = 15$ donc la valeur maximale de E est $E = 15 - E_{max} = 15 - 7 = 8$. Ensuite, la valeur maximale de la mantisse est obtenue pour tous les bits fixés à 1 :

$$m = 1 + \sum_{k=1}^{4} \frac{1}{2^k} = 1 + \frac{1}{2} + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} = \frac{1 - (\frac{1}{2})^5}{1 - \frac{1}{2}} = 2 \times (1 - (\frac{1}{2})^5) = 2 - 2^{-4},$$

et donc le nombre associé est $2^8(2 - 2^{-4}) = 2^8 - 2^4 = 256 - 16 = 140$.

b. Pour A : on a $s = 1$ (négatif), $e = 5$ et donc $E = 5 - 7 = -2$. La mantisse est donné par : $m = 1 + \frac{1}{2}$. Finalement, $A = -(1 + \frac{1}{2}) \times 2^{-2} = -\frac{1}{4} - \frac{1}{8} = -\frac{3}{8} = -0,375$.